To-Do List Manager Project Report

Task Class

```
import tkinter as tk
from tkinter import messagebox, simpledialog, filedialog
import pickle
from datetime import datetime

class Task:
    def __init__(self, description, priority=None, due_date=None):
        self.description = description
        self.priority = priority
        self.due_date = due_date
        self.completed = False

    def __str__(self):
        status = "[X]" if self.completed else "[ ]"
        priority = f" | Priority: {self.priority}" if self.priority else ""
        due = f" | Due: {self.due_date}" if self.due_date else ""
        return f"{status} {self.description}{priority}{due}
```

This class is used to define each task. It stores task description, priority, due date, and completion status.

Main Application Setup

```
class ToDoApp:
    def __init__(self, root):
        self.root = root
        self.root.title("To-Do List Manager")
        self.tasks = []

        # Listbox to show tasks
        self.listbox = tk.Listbox(root, width=60, height=15)
        self.listbox.pack(pady=10)

        # Buttons
        btn_frame = tk.Frame(root)
        btn_frame.pack()

        tk.Button(btn_frame, text="Add Task", command=self.add_task).grid(row=0, column=0, padx=5)
        tk.Button(btn_frame, text="Mark Complete", command=self.mark_complete).grid(row=0, column=1,
        tk.Button(btn_frame, text="Remove Task", command=self.remove_task).grid(row=0, column=2, padx
        tk.Button(btn_frame, text="Add Priority", command=self.add_priority).grid(row=1, column=0, pa
        tk.Button(btn_frame, text="Add Due Date", command=self.add_due_date).grid(row=1, column=1, pa
        tk.Button(btn_frame, text="Save Tasks", command=self.save_tasks).grid(row=1, column=2, padx=5
        tk.Button(btn_frame, text="Load Tasks", command=self.load_tasks).grid(row=1, column=3, padx=5
        tk.Button(btn_frame, text="Exit", command=root.quit).grid(row=2, column=0, padx=5)

        self.update_listbox()
```

This section sets up the main application window. It creates the listbox to display tasks and buttons for different actions like adding, removing, saving, and loading tasks.

Add Task Function

```
def add_task(self):
    desc = simpledialog.askstring("Add Task", "Enter task description:")
    if desc:
        task = Task(desc)
        self.tasks.append(task)
        self.update_listbox()
```

This function lets the user add a new task by entering a description.

Mark Complete Function

```
    def mark_complete(self):
        try:
            index = self.listbox.curselection()[0]
            self.tasks[index].completed = True
            self.update_listbox()
        except:
            messagebox.showwarning("Warning", "Select a task first!")
```

This function marks a selected task as completed.

Remove Task Function

```
    def remove_task(self):
        try:
            index = self.listbox.curselection()[0]
            del self.tasks[index]
            self.update_listbox()
        except:
            messagebox.showwarning("Warning", "Select a task first!")
```

This function removes the selected task from the list.

Add Priority Function

```
    def add_priority(self):
        try:
            index = self.listbox.curselection()[0]
            priority = simpledialog.askinteger("Add Priority", "Enter priority (1-5):")
            if priority and 1 <= priority <= 5:
                self.tasks[index].priority = priority
                self.update_listbox()
            else:
                messagebox.showerror("Error", "Priority must be between 1 and 5.")
        except:
            messagebox.showwarning("Warning", "Select a task first!")
```

This function allows the user to assign a priority level (1-5) to a selected task.

Add Due Date Function

```
    def add_due_date(self):
        try:
            index = self.listbox.curselection()[0]
            due = simpledialog.askstring("Add Due Date", "Enter due date (YYYY-MM-DD):")
            if due:
                try:
                    datetime.strptime(due, "%Y-%m-%d")  # validate format
                    self.tasks[index].due_date = due
                    self.update_listbox()
                except:
                    messagebox.showerror("Error", "Invalid date format! Use YYYY-MM-DD.")
        except:
            messagebox.showwarning("Warning", "Select a task first!")
```

This function lets the user add a due date to a selected task, ensuring the format is YYYY-MM-DD.

Save and Load Functions

```
    def save_tasks(self):
        file = filedialog.asksaveasfilename(defaultextension=".pkl")
        if file:
            with open(file, "wb") as f:
                pickle.dump(self.tasks, f)
            messagebox.showinfo("Saved", "Tasks saved successfully!")

    def load_tasks(self):
```

```
            file = filedialog.askopenfilename(filetypes=[("Pickle Files", "*.pkl")])
            if file:
                with open(file, "rb") as f:
                    self.tasks = pickle.load(f)
                self.update_listbox()
```
These functions save tasks to a file and load them back later.

Update Listbox Function

```
        def update_listbox(self):
            self.listbox.delete(0, tk.END)

            # Sort by priority then due date
            def sort_key(task):
                pr = task.priority if task.priority else 9999
                try:
                    due = datetime.strptime(task.due_date, "%Y-%m-%d") if task.due_date else datetime.max
                except:
                    due = datetime.max
                return (pr, due)

            for task in sorted(self.tasks, key=sort_key):
                self.listbox.insert(tk.END, str(task))
```
This function updates the listbox to display tasks sorted by priority and due date.

Run Application

```
        if __name__ == "__main__":
            root = tk.Tk()
            app = ToDoApp(root)
            root.mainloop()
```
This is the main entry point. It starts the Tkinter window and runs the application.